Before we dig deeper into tweaking the final look and adding more UI elements, let's think about one obvious problem. We will probably not just have one expense and definitely not an expense where the data is just hard coded into the Component. The date is hard coded here, the title is hard coded, the amount is hard coded and it's the only expense we have.

We want to have multiple ExpenseItem Components in action. And after all that is one of the core ideas behind Components. It's not just the separation of concerns, it's also the reusability. We want to define a Component and its HTML code once and then we want to be able to reuse it. And at the moment, we're not able to do that. But before that let's get rid of the hard coded values. Let's create a date variable.

Point to remember: Month in JavaScript starts with 0, so to add March we have to use number 2.

To output dynamic data, you use a special syntax React gives you inside of these JSX code snippets. You can replace hard coded data with opening and closing curly braces. And the special thing about these curly braces inside of your JSX code snippets is that in these curly braces between them, you can run basic JavaScript expressions.

For example, { 1 + 1 } you can execute something like this there and it's now the result of this expression which will be output on the screen. In place of an expression, we can also use the name of a variable inside curly braces. For date we have to keep in mind that it's a date object. And therefore, if we try to save it like this it just breaks. However, we can call ToISOString() for example a built in method available on all date objects to output it as a string.